Desarrollo Multiparadigma

Silvia Amaro, Pablo Quiroga, Luis Reynoso
{samaro, pquiroga, lreynoso}@uncoma.edu.ar
Claudio Vaucheret, Lidia López, Daniel Dolz
{vaucheret, lidiamlopez, daniel.dolz}@gmail.com
Andrea Granados, Ingrid Godoy, Viviana Sánchez, Maximiliano Klemen
{agranados80, ingriduncoma, sanchez.viviana, maximilianoklemen}@gmail.com
Paola Pérez
ppercat@hotmail.com

Facultad de Informática
Universidad Nacional del Comahue, Argentina

1. Resumen

En la búsqueda de modelos, técnicas y procesos para el desarrollo de la programación en un ambiente multiparadigma aparecen situaciones que producen una cierta evolución de los paradigmas clásicos dando respuesta a las mismas. Aparecen extensiones hacia la orientacióna agentes y a aspectos para dar respuesta a la complejidad que van presentando el desarrollo de sistemas a gran escala. En esta etapa se ha consolidado la idea de redefinir la programación como la elección de aquellos conceptos que permitan desarrollar la tarea del programador de manera apropiada donde resolver un problema en programación significa elegir los conceptos adecuados que permitan alcanzar la solución.
Palabras clave: Programación Multiparadigma, Lenguajes de Programación, Orientación a Agentes, Dominios Complejos.

2. Contexto

Está investigación se enmarca dentro del proyecto de investigación ”Técnicas Avanzadas y Análisis para el Desarrollo Multiparadigma”

3. Introducción

Un paradigma de programación provee y determina la visión y métodos que un programador utiliza en la construcción de un programa o subprograma. Diferentes paradigmas resultan en diferentes estilos de programación y en diferentes formas de pensar la solución de los problemas. En la ciencia de la computación, el término paradigma denota la esencia del proceso de desarrollo de software. También se puede considerar el término paradigma basado en la perspectiva de lenguaje de programación. Esta percepción de paradigma se puede comprobar en , donde Coplien identifica partes comunes y variantes que en conjunto constituyen un paradigma. Un paradigma es entonces una configuración de concordancias y variaciones. Mientras que los paradigmas clasificados como básicos son los más representativos no constituyen todos los paradigmas existentes, existen extensiones y variantes de éstos. A continuación se detallan paradigmas basados en los conceptos de los paradigmas básicos: Ningún paradigma es capaz de resolver todos los problemas de forma sencilla y eficiente, por lo tanto es

útil poder elegir entre distintos .estilos"de programación dependiendo del tipo de problema. También hay lenguajes que permiten mezclar los paradigmas que, en principio, parecerían irreconciliables . Por ejemplo, el lenguaje Oz emplea programación lógica, funcional, orientada a objeto y otras [8]. Lenguajes como C++ combinan el paradigma imperativo y el orientado a objetos. El lenguaje Leda soporta los paradigmas imperativo, orientado a objetos, lógico y funcional [16]. El lenguaje Python combina el paradigma imperativo, funcional, orientado objetos y orientado a aspectos. El diseño de sistemas que desempeñan tareas críticas en ambientes dinámicos complejos se ha vuelto de suma relevancia en los tiempos que corren. Para hacer frente a la complejidad inherente a la construcción exitosa de estos sistemas, las distintas áreas de las ciencias de la computación han elaborado metodologías y herramientas que ayudan en todas las etapas del ciclo de desarrollo. En este contexto es que surge, a principios de la década de 1990, a raíz de varios trabajos precedentes, un nuevo paradigma de diseño y programación, denominado paradigma orientado a agentes . Bajo esta perspectiva, los sistemas son vistos como un conjunto de entidades independientes y autónomas que colaboran entre ellas para alcanzar un objetivo en común. Si bien en principio puede verse como una analogía de la orientación a objetos, la diferencia radica sobre todo en que las entidades componentes del sistema actúan proactivamente, persiguiendo objetivos individuales concretos [6]. Más aún, el comportamiento de estas entidades puede ser descripto en función de su estado mental y las acciones que lleva a cabo a raíz de dicho estado.

Actualmente existen lenguajes de programación de agentes de diversas características. Algunos están fuertemente basados en fundamentos teóricos formales (AGENT0 , GOLOG , FLUX), mientras que otros son extensiones de un lenguaje existente, sin una base teórica completamente formada. Algunos ofrecen construcciones básicas para definir la arquitectura interna de los agentes (3APL , Jadex), mientras que otros se abstraen de estos detalles, enfocándose en otras características como la comunicación entre agentes o la movilidad. Estas diferencias tan marcadas muchas veces provocan que un sistema que originalmente se diseñó utilizando las herramientas que brinda el paradigma orientado a agentes termine siendo implementado con lenguajes de programación más tradicionales, pero de probada madurez y utilidad.

La lógica de los aspectos se entrelaza en la aplicación destino, y en esta situación múltiples aspectos, desarrollados de forma independiente pueden producir un comportamiento inesperado en el sistema. La programación orientada a aspectos en general se ha presentado como extensión de la orientación a objetos, sin embargo resulta de interés analizar el beneficio que pueda producir la aplicación de conceptos de orientación a aspectos en lenguajes no orientados a objetos y multiparadigma.

4. Líneas de Investigación y Desarrollo

El objetivo general es establecer modelos, procesos y técnicas para mejorar el desarrollo de sistemas en un ambiente multiparadigma. Tendientes a lograr tal objetivo se proponen diversas líneas de trabajo, que se orientan a lograr algunos objetivos particulares .

- Realizar una expansión sintáctica que permita simular las variables globales y estructuras de control imperativo en un lenguaje multiparadigma que contempla los paradigmas lógico, funcional, orientado a objetos, orientado a accesos, con restricciones y literate.utilizado; y extender el análisis para el caso multivariante. Se está trabajando sobre el lenguaje Ciao Prolog un lenguaje multiparadigma basado en expansiones sintácticas que utiliza la técnica de compilación de control para implementar los distintos paradigmas [7].

- Definir e implementar un lenguaje de programación orientado a agentes con al menos las siguientes características: - posibilidad de dotar a los agentes con capacidades cognitivas de alto nivel, a través

de una adaptación del método de programación FLUX [6] , - flexibilidad para la definición de una arquitectura interna, permitiendo construir desde un agente con un control principal muy simple hasta agentes basados en la arquitectura BDI . El lenguaje se implementará como una expansión del lenguaje de programación multiparadigma Ciao Prolog [7, 2], para aprovechar las facilidades de expansión que brinda el mismo, la madurez de su desarrollo y de sus herramientas de programación y debbugging. Esta línea de trabajo pretende impulsar el establecimiento de una plataforma para programación de agentes inteligentes que se consolide como una herramienta de gran utilidad para el desarrollo.

- En el contexto de la orientación a aspectos, analizar los resultados que se pueden obtener al tomar paradigmas anfitriones distintos del orientado a objetos para considerar las características de la orientación a aspectos. El desarrollo orientado a aspectos se presenta apropiado para asistir al desarrollador en aislar puntos de variación en un programa. Esto ayuda al programa a evolucionar y adaptarse a nuevos requerimientos de cambio durante el ciclo de vida del programa. En particular se está considerando la programación orientada a Aspectos en el contexto del desarrollo orientado a servicios en general y servicios web en particular. Se está realizando una comparación entre las distintas iniciativas existentes respecto de utilizar aspectos en combinación con los lenguajes de descripción de workflows tipo BPEL4WS. Además se está considerando las distintas posibilidades de la incorporación de aspectos al desarrollo basado en componentes.[13, 14, 15]

- Redefinir la programación como la elección de aquellos conceptos que permitan desarrollar la tarea del programador de manera apropiada. Resolver un problema en programación significa elegir los conceptos adecuados que permitan alcanzar la solución. Estos conceptos le brindan al programador una expresividad esencial para desarrollar su tarea ya que definen a los lenguajes de programación. La elección del lenguaje de programación debe estar apoyada en los conceptos que soporta para dar una respuesta adecuada [12].

5. Resultados esperados

El desarrollo multiparadigma permite la construcción de sistemas mediante la selección del paradigma apropiado en cada situación. El producto de la presente propuesta puede ser de utilidad para el desarrollo de aplicaciones en dominios particulares y que deban cumplir con propiedades particulares, tales como eficiencia, seguridad y calidad, brindando apoyo al desarrollador para la evaluación y selección de las mejores técnicas y paradigmas para la construcción de sistemas en gran escala.

6. Formación de Recursos Humanos

El mayor impacto del presente proyecto se centra en la formación de recursos humanos, consolidación de grupos de investigación e interacción entre grupos interdisciplinarios. Parte de los autores están dando sus primeros pasos en investigación. Actualmente se están desarrollando en el contexto del proyecto 3 tesinas de grado, cuya finalización está prevista para el segundo semestre del corriente año. Asimismo se están desarrollando 1 tesis de magister y 2 tesis doctorales y se prevee la iniciación de nuevos estudios de posgrado.

Referencias

[1] Guido Soldner and Rudiger Kapetza. *AOCI: An Aspect-Oriented Component Infraestructure*. Workshop on Component Oriented Programming WCOP, ECOOP 2007, Berlin, Alemania.

[2] C. Vaucheret and F. Bueno. *More precise yet efficient type inference for logic programs.* In International Static Analysis Symposium, LNCS 2477, pp 102-116. Springer-Verlag, September 2002.

[3] Éric Tanter and Jacques Noyé. *A Versatile Kernel for Multi-Language AOP.* In Proceedings of the ACM SIGPLAN/SIGSOFT Conference on Generative Programming and Component Engineering (GPCE 2005), LNCS, Springer-Verlag, Tallin, Estonia, September, 2005.

[4] K. Davis and J. Striegnitz. *Multiparadigm Programming in Object-Oriented Languages: Current Research*. In Patrick Eugster (eds.): Object-Oriented Technology, LNCS 5475, Programming and Software Engineering, pp 104-115, 2009.

[5] F. Spiessens and P. Van Roy. *The Oz-E PRoject: Design Guidelines for a Secure Multiparadigm Programming Language.*. http://www.info.ucl.ac.be/ pvr/oze.pdf

[6] M. Thielscher. *FLUX: A logic programming method for reasoning agents*. In Theory and Practice of Logic Programming, Special Issue on Constraint Handling Rules, 5(4&5):533-565, 2005.

[7] M.V. Hermenegildo, F. Bueno, M. Carro, P. López, J.F. Morales, and G. Puebla. *An overview of the ciao multiparadigm language and program development environment and its design philosophy*. LNCS 5065, Concurrency, Graphs and Models. pp 209-237, 2008.

[8] *The Oz 1.1 Programming System*. Available at http://www.ps.uni-sb.de/oz1/

[9] J. O. Coplien. *MultiParadigm Design*. Vrije Universiteit Brussel, Faculteit Wetenschanppen - Departement Informatica.

[10] P. Van Roy. *Programming Paradigms for Dummies: What Every Programmer Should Know* Department of Computing Science and Engineering - Université catholique de Louvain (UCL) à Louvain-la-Neuve. Available at http://www.info.ucl.ac.be/ pvr/VanRoyChapter.pdf

[11] P. Van Roy, P. Brand, D. Duchier, S. Haridi, M. Henz and C. Schulte. *Logic Programming in the Context of Multiparadigm Programming: the Oz Experience*. Available at http://www.info.ucl.ac.be/ pvr/tutFinal.pdf

[12] Narbel. *Functional Programming at work in Object-Oriented Programming*. In Journal of Object Technology, vol 8, no 6., pp 181-209. Septiembre-Octubre 2009. Available at http://www.jot.fm/issues/issue_2009_07/column5.

[13] Mathieu Braem, Kris Verlaenen, Niels Joncheere, Wim Vanderperren, Ragnhild Van Der Straeten, Eddy Truyen, Wouter Joosen, and Viviane Jonckers. *Isolating Process-Level Concerns using Padus*. In Proc. of the 4th International Conference on Business Process Management (BPM), volume 4102 of LNCS, pages 113-128. Springer, September 2006.

[14] Ying Zhang, Hao Yang, Junliang Chen, Xiangwu Meng. *Themes4BPEL: An Efficient Aspect-Oriented Web Service Composition Design Approach*.In Proc. of the 11th International Conference on Advanced Communication Technology, volume 1 ACM, 2009.

[15] Yang Xu, Shengqun Tang, Youwei X, Zukai Tang. *Towards Aspect Oriented Web Sevice Composition with UML*. Im Proc. of the 6th IEEE/ACIS International Conference on Computer and Information Science, pages 279-284, Julio 2007.

[16] T. Budd. *Multiparadigm Programming in Leda*. Addison Wesley 1995. ISBN 0-201-82080-3.

[17] M. Fowler, *Domain Specific Languages* work in progress, available at http://www.martinfowler.com/dslwip/